HW1 Programming Problem 2 (10 points)

In this problem, we are given a function $L(w_1, w_2)$ with a known functional form. You will perform gradient descent to find a global minimum. The goal is to find what initial guesses and learning rates (step sizes) lead the algorithm to find the global minimum.

The function $L(w_1, w_2)$ is defined as:

$$L(w_1, w_2) = \cos(4w_1 + w_2/4 - 1) + w_2^2 + 2w_1^2$$

A Python function for `L(w_1, w_2)` is given.

Gradients

First, we must define a gradient of L. That is $\nabla L = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}\right]$. First, compute these derivatives by hand. Then, in the cell below, complete the functions for the derivatives of `L` with respect to `w1` and `w2` .

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt

def L(w1, w2):
    return np.cos(4*w1 + w2/4 - 1) + w2*w2 + 2*w1*w1

def dLdw1(w1, w2):
    # YOUR CODE GOES HERE
    return (4*(np.sin(-4*w1 - w2/4 + 1))+4*w1)


def dLdw2(w1, w2):
    # YOUR CODE GOES HERE
    return ((np.sin(-4*w1 - w2/4 + 1)*(1/4))+2*w2)
```

Gradient Descent

The function `plot_gd` performs gradient descent by calling your derivative functions. Take a look at how this works. Then, run the interactive gradient descent cell that follows and answer the questions below.

In [14]:
```
def plot_gd(w1, w2, log_stepsize, log_steps):
    stepsize = 10**log_stepsize
    steps = int(10**log_steps)

    # Gradient Descent
    w1s = np.zeros(steps+1)
    w2s = np.zeros(steps+1)

    for i in range(steps):
        w1s[i], w2s[i] = w1, w2
        w1 = w1 - stepsize * dLdw1(w1s[i],w2s[i])
        w2 = w2 - stepsize * dLdw2(w1s[i],w2s[i])
    w1s[steps], w2s[steps] = w1, w2

    # Plotting
    vals = np.linspace(-1,1,50)
    x, y = np.meshgrid(vals,vals)
    z = L(x,y)

    plt.figure(figsize=(7,5.8),dpi=120)
    plt.contour(x,y,z,colors="black", levels=np.linspace(-.5,3,6))
    plt.pcolormesh(x,y,z,shading="nearest",cmap="Blues")
    plt.colorbar()

    plt.plot(w1s,w2s,"g-",marker=".",markerfacecolor="black",markeredgecolor="
    plt.scatter(w1s[0],w2s[0],zorder=100, color="blue",marker="o",label=f"$w_0
    plt.scatter(w1,w2,zorder=100,color="red",marker="x",label=f"$w^*$ = [{w1:.
    plt.legend(loc="upper left")

    plt.axis("equal")
    plt.box(False)
    plt.xlabel("$w_1$")
    plt.ylabel("$w_2$")
    plt.xlim(-1,1)
    plt.ylim(-1,1)
    plt.title(f"Step size = {stepsize:.0e}; {steps} steps")
    plt.show()
```

In [15]:
```
%matplotlib inline
from ipywidgets import interact, interactive, fixed, interact_manual, Layout,

slider1 = FloatSlider(
    value=0,
    min=-1,
    max=1,
    step=.1,
    description='w1 guess',
    disabled=False,
    continuous_update=True,
    orientation='horizontal',
    readout=False,
    layout = Layout(width='550px')
)

slider2 = FloatSlider(
    value=0,
    min=-1,
    max=1,
    step=.1,
    description='w2 guess',
    disabled=False,
    continuous_update=True,
    orientation='horizontal',
    readout=False,
    layout = Layout(width='550px')
)

slider3 = FloatSlider(
    value=-1.5,
    min=-3,
    max=0,
    step=.5,
    description='step size',
    disabled=False,
    continuous_update=True,
    orientation='horizontal',
    readout=False,
    layout = Layout(width='550px')
)

slider4 = FloatSlider(
    value=2,
    min=0,
    max=3,
    step=.25,
    description='steps',
    disabled=False,
    continuous_update=True,
    orientation='horizontal',
    readout=False,
    layout = Layout(width='550px')
)


interactive_plot = interactive(
```

```
    plot_gd,
    w1 = slider1,
    w2 = slider2,
    log_stepsize = slider3,
    log_steps = slider4,
    )
output = interactive_plot.children[-1]
output.layout.height = '620px'

interactive_plot
```

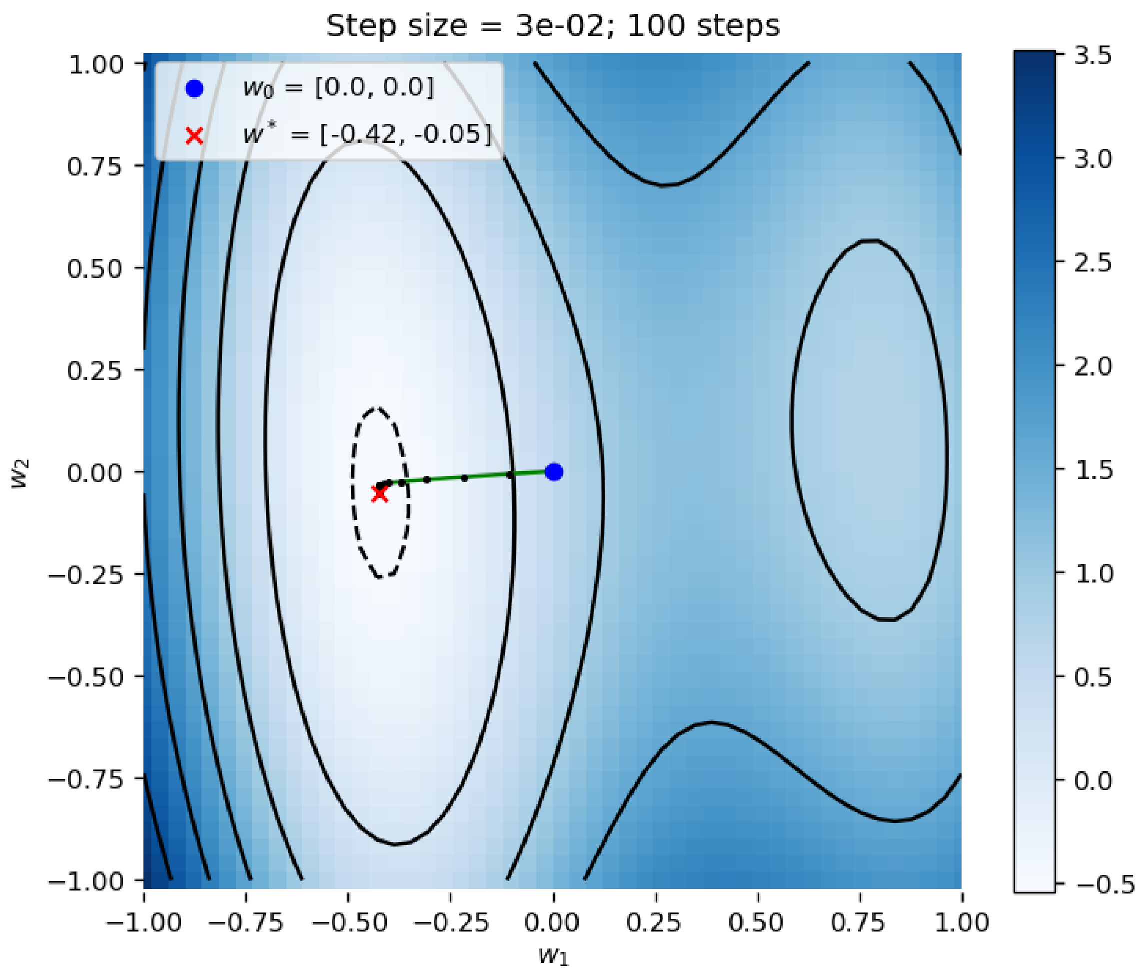
Out[15]:

w1 guess

w2 guess

step size

steps



Questions

Play around with the sliders above to get an intuition for which initial conditions/learning rates lead us to find the global minimum at [-0.42, -0.05]. Then answer the following questions:

1. Set w_0 to [0.2, 0.8] and step size to 1e-01. After 100 steps of gradient descent, what w^* do we reach?
2. Keep parameters from the previous question, but change the initial guess to [0.3, 0.8]. Now what is the optimum we find?
3. Set w_0 to [-1.0, -1.0] and number of iterations to 1000 and step size to 1e-03. What w^* do we reach, and why is it not exactly the global minimum?
4. In general, what happens if we set learning rate too large?

For global minimum at [-0.42,-0.05], the initial conditions are [0.2,0.8] with step size as 1e-01 and 100 steps

1. w* = [-0.42, -0.05]
2. Optimum = [0.80,0.10]
3. w* = [-0.42, -0.18] The function takes longer time to converge with smaller step size which can lead to an error. There is a need for higher step size or higher iterations to reach the global minima.
4. If the learning rate is too large then the oscillations start happening which therefore leads to errors. The function misses the global minimum.

```
In [ ]:
```